**3 Reasons Angular Rocks**

[Angular.js](https://angularjs.org/) is an awesome framework for medium-complexity, single-page web applications. Its feature set and use cases fit perfectly in the complexity gap between Backbone.js and Ember.js.

Composed of many interesting and useful features such as animation and touch interaction, Angular provides an architecture that makes your single-page web app a joy to work with.

**1. Promises, Promises, Promises**

[Promises](http://wiki.commonjs.org/wiki/Promises) are nice, and they help to avoid a common problem in JavaScript: [callback hell](http://callbackhell.com/). [Angular](https://spin.atomicobject.com/tag/angular-js/) provides the [$q](https://docs.angularjs.org/api/ng/service/$q) service, It turns a common Backbone.js site:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | doThing1("a", function(thing1) {  doThing2("b", function(thing2) {  doThing3("c", function(thing3) {  console.log("I am the triangle of doom", thing3)  })*;*  })*;*  })*;* |

into elegant chains of functions in Angular:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | doThing1().  then(function(thing1) {...}).  then(function(thing2){...}).  then(function(thing3) {...}).  catch(function(error) {...}).  final(function(){...}) |

Promises are spread throughout the framework, most importantly in network access and the router.

**2. HTTP and REST Made Easy**

Angular, similar to [Ember](https://spin.atomicobject.com/2013/12/02/emberjs-abstraction/), is built up on layers of [abstractions](https://spin.atomicobject.com/2013/04/14/effortless-abstraction-ember/). The $httpservice makes hitting a URL easy, and also utilizes promises. If we wanted to do a POST to the /contact URL using $http, it would look like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $http({method: 'POST', url: '/contact', params: {name: "Joe", email: "joe@foob.ar"}).  success(**function**(data, status, headers, config) {  *// do something with our response*  }).  error(**function**(data, status, headers, config) {  *// process the error response*  }); |

Building up on top of [$http](https://docs.angularjs.org/api/ng/service/$http) is [$resource](https://docs.angularjs.org/api/ngResource/service/$resource), which makes RESTful API interactions a lot thinner than if they were implemented with $http. Retrieving a contact by id would look as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **var** Contact = $resource('/contact/:id', {id:'@id'});  **var** joe = Contact.**get**({id:42}).$promise.  then(**function**(data) {  *// process contact #42*  }).  **catch**(**function**(response) {  *// process the error response*  }) |

Building up a further layer of abstraction is [Restangular](https://github.com/mgonto/restangular). While not maintained by the Angular core community, Restangular is well-documented and makes RESTful interactions even simpler than $resource. Transforming our contact example to utilize Restangular would look as follows:

|  |  |
| --- | --- |
| 1  2  3 | Restangular.one('contact', 42).  then(**function**(data){...}).  **catch**(**function**(response){...}) |

Restangular provides caching via ETags for free, easy nested resources and the ability to quickly generate a URL or specify your own. Much like everything else so far, Restangular always returns a promise. This makes resolved Restangular resources easy to inject into controllers using Angular router’s resolve.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | $routeProvider.when('/contacts/:id', {  templateUrl: template\_path("contact\_details.html"),  controller: "ContactCtrl",  resolve: {  contact: ["$route", "Restangular", **function**($route, Restangular) {  *// contact resource with :id is injected into ContactCtrl when resolved*  **return** Restangular.one('contacts', $route.current.params.id).**get**();  })]  }); |

**3. Dependency Injection – Angular’s Foundation**

[Angular’s dependency injection](https://docs.angularjs.org/guide/di) capabilities are by far my favorite functionality built into Angular. The entirety of the framework, as demonstrated above with the $routeProvider’s resolve capabilities, is structured around injection. Angular’s dependency injection has also inspired other projects, such as intravenous in the node ecosystem.

[Dependency injection](https://spin.atomicobject.com/2012/10/11/first-foray-dependency-injection/) makes the Angular framework and single page web apps built on it extremely modular and testable. Dependency injection is structure into 2 components, the objects being injected, and the objects which have their dependencies injected into them.

The objects being injected are known as values, constants, factories, services and providers, ordered from least complex to most. I won’t go into too much detail about the differences since each one is [well documented](https://docs.angularjs.org/api/auto/object/$provide) by the Angular team. The following code defines an Authentication factory which can be injected into other objects.

|  |  |
| --- | --- |
| 1  2  3  4  5 | myModule.factory("Authentication", **function**() {  **return** {  login: **function**(user, password) {...}  };  }); |

Lets modify our Authentication factory to have Restangular injected into it.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | myModule.factory("Authentication", **function**(Restangular) {  **return** {  login: **function**(user, password) {  **return** Restangular.one("/login").post({user: user, password: password});  }  };  }); |

The complication here is that since angular’s [dependency resolution](https://docs.angularjs.org/api/auto/service/$injector) is based off of the name of our argument, minification will cause our dependency name to be lost. This can be resolved in a variety of ways but the most favored, due to it’s ability to be inlined, is as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | myModule.factory("Authentication", ["Restangular", **function**(Restangular) {  **return** {  login: **function**(user, password) {  **return** Restangular.one("/login").post({user: user, password: password});  }  };  }]); |